Implementasi Algoritma Rabin Karp untuk Pendeteksian Plagiat Dokumen Teks Menggunakan Konsep *Similarity*

Salmuasih
Teknik Informatika
STMIK AMIKOM Yogyakarta
Yogyakarta, Indonesia
email: sal_252@yahoo.com

Andi Sunyoto
Teknik Informatika
STMIK AMIKOM Yogyakarta
Yogyakarta, Indonesia
email: andi@amikom.ac.id

Abstract*—Plagiarism is a crime, because it recognizes the work of others as personal work. Plagiarism is becoming serious problem as the rapid development of technology. Ironically acts of plagiarism is most prevalent among academics such as students and learners. In this case, the most often done is copy-paste-edit text documents. This study aims to analyze the act of plagiarism based on similarity contents of the document, such as copy&paste and disguised plagiarism. We implement strategic approach of Rabin-Karp algorithm, which is a multiple-pattern search algorithm. This algorithm is one of the best in analyzing compatibility between documents. The use of hashing techniques make the matching algorithm more efficient because it will just compare multiple digit of numbers. The final result is the percentage of similarity between documents being tested.

Keywords*—Rabin Karp algorithm; plagiarism; similarity; string matching; hashing.

I. PENDAHULUAN

Plagiat selalu menjadi sorotan, terutama di sektor akademis. Sering ditemui pekerjaan teman yang dihargai dengan nilai "0" karena diketahui telah menyalin hasil pekerjaan temannya yang lain. Praktik menyalin beberapa bagian atau keseluruhan tulisan menjadi hal biasa yang sering ditemukan pada penelitian.

Pencegahan dan pendeteksian dini merupakan cara yang dapat dilakukan untuk mengurangi plagiat. Pencegahan berarti menghalangi munculnya plagiat yang lebih ditekankan kepada moral masyarakat dan sistem pendidikan. Cara ini akan memberikan efek jangka panjang.

Beberapa perangkat lunak yang didesain untuk mendeteksi plagiat dokumen, diantaranya Turnitin, Eve2, CopyCatchGold, WordCheck, Glatt, Moss, JPlag. Berdasarkan analisis informasi yang ada di web, pendeteksi terbaik sesuai fungsinya adalah Turnitin [1].

Duplikasi dokumen, deteksi plagiat, dan pencocokan *string* telah banyak dibahas pada penelitian-penelitian sebelumnya. Algoritma yang digunakan diantaranya Winnowing, Smith Waterman, Boyer Moore, dan Rabin Karp. Namun sebagian besar tanpa menggunakan *preprocessing*, sehingga berpengaruh pada akurasi *similarity.*

Pendeteksian plagiat menggunakan konsep *similarity* atau kemiripan dokumen merupakan salah satu cara untuk mendeteksi *copy&paste plagiarism* dan *disguised plagiarism*. Menggunakan algoritma Rabin Karp yang menerapkan metode *fingerprinting* yang hanya melakukan pencocokan pola *string*, pendeteksian plagiat ini tidak memperhatikan adanya penulisan sumber rujukan.

Pada sistem deteksi ini akan diaplikasikan *text mining* untuk tahap *preprocessing* dan algoritma Rabin Karp untuk *string matching*. Algoritma Rabin Karp adalah algoritma *multiple pattern search* yang sangat efisien untuk mencari *string* dengan pola banyak. Selanjutnya akan dibahas bagaimana algoritma Rabin Karp bekerja sekaligus implementasinya pada sebuah aplikasi dalam mendeteksi plagiat.

Algoritma Rabin Karp yang digunakan disini adalah hasil modifikasi oleh [2] dan telah dibandingkan keakuratannya dalam pencocokan *string* bahwa algoritma Rabin Karp sesudah modifikasi waktu prosesnya lebih baik dibandingkan sebelum modifikasi.

II. METODE PENELITIAN

A. *Plagiarisme*

Plagiarisme atau plagiat adalah penjiplakan atau pengambilan karangan, pendapat orang lain dan menjadikannya seolah-olah karangan sendiri [3]. Pendekatan deteksi plagiat terbagi menjadi *intrinsic* dan *external*. Pendekatan *external* terbagi lagi menjadi tiga, yaitu perbandingan teks lengkap, kesamaan kata kunci dan

fingerprinting. Perbandingan teks lengkap diterapkan untuk membandingkan semua isi dokumen, kesamaan kata kunci bekerja dengan cara mengekstrak dan membandingkan kata kunci antardokumen, dan *fingerprinting* untuk mendeteksi kemiripan antardokumen dengan prinsip *hashing*.

Berdasarkan penelitian [4] mengkategorikan praktek plagiat berdasarkan cara yang digunakan, diantaranya :

1. *Copy&Paste plagiarism*, menyalin setiap kata tanpa perubahan.
2. *Disguised plagiarism*, tergolong kedalam praktek menutupi bagian yang disalin, teridentifikasi ke dalam empat teknik, yaitu s*hake&paste*, e*xpansive plagiarism, contractive plagiarism,* dan *mosaic plagiarism.*
3. *Technical disguise,* teknik meringkas untuk menyembunyikan konten plagiat dari deteksi otomatis dengan memanfaatkan kelemahan dari metode analisis teks dasar, misal dengan mengganti huruf dengan simbol huruf asing.
4. *Undue paraphrasing,* sengaja menuliskan ulang pemikiran asing dengan pemilihan kata dan gaya plagiator dengan menyembunyikan sumber asli.
5. *Translated plagiarism,* mengkonversi konten dari satu bahasa ke bahasa lain.
6. *Idea plagiarism,* menggunakan ide asing tanpa menyatakan sumber.

Self plagiarism, penggunaan sebagian atau keseluruhan tulisan pribadi yang tidak dibenarkan secara ilmiah.

B. Similarity

1) Similarity Dokumen

Konsep *similarity* sudah menjadi isu yang sangat penting di hampir setiap bidang ilmu pengetahuan. [5] dalam disertasinya menjelaskan tiga macam teknik yang dibangun untuk menentukan nilai *similarity* (kemiripan) dokumen.

a) *Distance-based similarity measure*

Distance-based similarity measure mengukur tingkat kesamaan dua buah objek dari segi jarak geometris dari variabel-variabel yang tercakup di dalam kedua objek tersebut. Metode *Distance-based similarity* ini meliputi *Minkowski Distance*, *Manhattan*/*City block distance*, *Euclidean distance*, *Jaccard Distance*, *Dice's Coefficient*, *Cosine similarity*, *Levenshtein Distance*, *Hamming Distance*, dan *Soundex distance*.

b) *Feature-based similarity measure*

Feature-based similarity measure melakukan penghitungan tingkat kemiripan dengan merepresentasikan objek ke dalam bentuk *feature-feature* yang ingin diperbandingkan. *Feature-based similarity measure* banyak digunakan dalam melakukan pengklasifikasian atau *pattern maching* untuk gambar dan teks.

c) *Probabilistic-based similarity measure*

Probabilistic-based similarity m*easure* menghitung tingkat kemiripan dua objek dengan merepresentasikan dua set objek yang dibandingkan dalam bentuk *probability*. Kullback Leibler Distance dan Posterior Probability termasuk dalam metode ini.

C. Pengukuran Nilai Similarity

Mengukur *similarity* (kemiripan) dan jarak antara dua entitas informasi adalah syarat inti pada semua kasus penemuan informasi, seperti pada *Information Retrieval* dan *Data Mining* yang kemudian dikembangkan dalam bentuk aplikasi, salah satunya adalah sistem deteksi plagiat.

Penggunaan ukuran *similarity* yang tepat tidak hanya meningkatkan kualitas pilihan informasi tetapi juga membantu mengurangi waktu dan biaya proses [5]. [6] menyarankan untuk mengaplikasikan *Dice's Similarity Coefficient* dalam penghitungan nilai *similarity* yang menggunakan pendekatan *k-gram.*

$$S = \frac{K * C}{(A + B)}$$

Dimana S adalah nilai similarity, A dan B adalah jumlah dari kumpulan *k-grams* dalam teks 1 dan teks 2. C adalah jumlah dari *k-grams* yang sama dari teks yang dibandingkan.

D. Text Mining

Text mining adalah salah satu bidang khusus dari *data mining*. [7] mendefinisikan *text mining* sebagai suatu proses menggali informasi dimana seorang *user* berinteraksi dengan sekumpulan dokumen menggunakan *tools* analisis yang merupakan komponen-komponen dalam *data mining*. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen.

Dalam memberikan solusi, *text mining* mengadopsi dan mengembangkan banyak teknik dari bidang lain, seperti Data mining, *Information Retrieval (IR)*, *Statistic and Mathematic, Machine Learning*, *Linguistic*, *Natural Languange Processing (NLP)*, dan *Visualization*. Kegiatan riset untuk *text mining* antara lain ekstraksi dan penyimpanan teks, *preprocessing* akan konten teks, pengumpulan data statistik dan *indexing*, dan analisa konten [8]. Tahapan dalam *text mining* meliputi *tokenizing*, *filtering*, *stemming*, *tagging* dan *analyzing* [9].

E. Algoritma Rabin Karp

String matching atau pencocokan *string* adalah subjek yang penting dalam kaitannya dengan *text-processing*. Penggunaan *string matching* mencakup pencarian pola dalam DNA *sequence*, *search engine* internet, menemukan halaman web

yang relevan pada *query*, dapat pula dimanfaatkan untuk mendeteksi adanya plagiarisme karya tulis. Termasuk dalam algoritma *string matching* diantaranya algoritma Naive, algoritma Rabin Karp, algoritma Finite Automaton, dan algoritma Knuth Morris Pratt [10].

Algoritma Rabin Karp ditemukan oleh Michael O. Rabin dan Richard M. Karp. Algoritma ini menggunakan metode *hash* dalam mencari suatu kata. Teori ini jarang digunakan untuk mencari kata tunggal, namun cukup penting dan sangat efektif bila digunakan untuk pencarian jamak [11].

F. Prinsip Kerja Algoritma Rabin Karp

Rabin Karp merepresentasikan setiap karakter ke dalam bentuk desimal digit (*digit radix-d*) $\sum = \{0, 1, 2, 3, \ldots, d\}$, dimana $d = |\sum|$. Sehingga didapat masukan *string k* berturut-turut sebagai perwakilan panjang *k* desimal. Karakter *string* 31415 sesuai dengan jumlah desimal 31,415. Kemudian pola *p* di-*hash* menjadi nilai desimal dan *string* direpresentasikan dengan penjumlahan *digit-digit* angka menggunakan aturan Horner's, misal [12]:

{ A, B, C, ..., Z } → { 0, 1, 2, ..., 26 }
• BAN → 1 + 0 + 13 = 14
• CARD → 2 + 0 + 17 + 3 = 22

Untuk pola yang panjang dan teks yang besar, algoritma ini menggunakan operasi *mod*, setelah dikenai operasi *mod q*, nilainya akan menjadi lebih kecil dari *q*.

Tetapi tidak semua nilai *hash* yang cocok berarti polanya cocok. Hal ini sering terjadi pada beberapa kasus, ini disebut *spurious hits.* Kemungkinan terjadinya diantaranya karena:

- Operasi *mod* terinterfensi oleh keunikan nilai *hash* (nilai *mod q* biasanya dipilih bilangan prima sehingga 10*q* hanya cocok dengan 1 kata komputer)
  14 mod 13 = 1
  27 mod 13 = 1
- Informasi hilang setelah penjumlahan
  BAN → 1 + 0 + 13 = 14
  CAM → 2 + 0 + 12 = 14

Sedangkan *pseudocode* dan rumus matematis yang digunakan adalah sebagai berikut [10]:

```
RABIN-KARP-MATCHER (T, P, d, q)
n  = T.length
m  = P.length
h  = dm-1 mod q
p  = 0
t0 = 0
for i = 1 to m                          // preprocessing
    p = (dp + P[i]) mod q
    t0 = (dt0 + T[i]) mod q
for s = 0 to n – m
   if p == ts                           // matching
   if P[1 .. m] == T[s + 1 .. s + m]
      print “Pattern occurs with shift” s
```

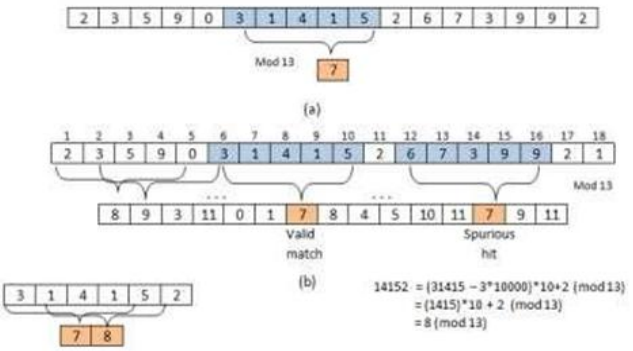
```
if s < n – m
    ts+1 = (d(ts - T[s + 1] h) + T[s + m + 1]) mod q
```

Rumus matematis:

$$t_{s+1} = (d\,(t_s - T[s+1]\,h) + T[s+m+1] \bmod q$$

dimana

t_s = nilai desimal dengan panjang *m* dari *substring T* [*s* + 1 .. *s* + *m*], untuk *s* = 0, 1, ..., *n* - *m*
t_{s+1} = nilai desimal selanjutnya yang dihitung dari t_s
d = *radix* desimal (bilangan basis 10)
h = d^{m-1}
n = panjang teks
m = panjang pola
q = nilai *modulo*



Gambar 1. Algoritma Rabin Karp

Pengurangan dengan *T*[*s*+1]**h* adalah untuk menghilangkan *high-order digit* dari t_s, mengalikan hasilnya dengan 10 untuk menggeser satu digit angka ke kiri, dan menambahkan *low-order digit* dengan *T* [*s* + *m* + 1]. Misalnya, jika *m* = 5 dan t_s = 31415, maka kita ingin menghapus *high-order digit T* [*s* +1] = 3, masukkan *low-order* digit baru (anggap *T* [*s* +5 +1] = 2) dan *modulo* = 3 untuk memperoleh

t_{s+1} = (10 (31415 - 3 * 10000)) + 2 *mod* 13
= 14152 *mod* 13
= 8

G. Peningkatan Performa Algoritma Rabin Karp

Telah dipahami bahwa spurious hit adalah beban tambahan bagi algoritma yang akan meningkatkan waktu proses. Hal ini dikarenakan algoritma harus membandingkan pola terhadap teks yang hasil modulonya sama tetapi nilai *hash*-nya berbeda. Untuk menghindari pencocokan yang tidak perlu, [2] memberikan solusi untuk tidak hanya membandingkan sisa hasil bagi, tetapi membandingkan hasil baginya juga.

REM (n1/q) = REM (n2/q)
and
QUOTIENT (n1/q) = QUOTIENT (n2/q)

Jadi, *successful hit* harus memenuhi dua syarat, yaitu nilai sisa hasil bagi dan nilai hasil baginya harus sama. Selebihnya adalah *unsuccessful hit* tanpa perlu melakukan pencocokan lagi. Hal ini berarti tidak ada pemborosan waktu untuk mengecek *spurious hit*.

```
RB_IMPROVED (T, P, d, q)
n   = T.length
m   = P.length
h   = dm-1 mod q
p   = 0
t0  = 0
Q = 0
pq = 0
h = dm-1  mod q
for i = 1 to m                                  // preprocessing
    p = (dp + P[i]) mod q
    t0 = (dt0 + T[i]) mod q
for s = 0 to n – m
   Q=T{s+1.....s+ m] div q
   If ( ts = p and Q = pq)                      // matching
      print “Pattern occurs with shift” s
   if s < n – m
      ts+1 = (d(ts - T[s + 1] h) + T[s + m + 1]) mod q
```

H. Perancangan Proses

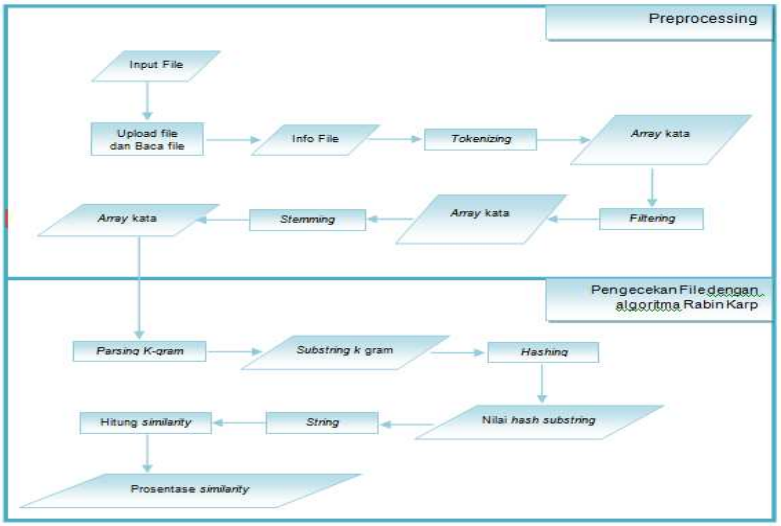
Sistem deteksi plagiat secara umum dirancang untuk dapat mendeteksi kemiripan isi pada dokumen teks, yang dimungkinkan kemiripan ini adalah hasil plagiat. *Input* sistem diperoleh dari *file*/dokumen (dalam hal ini *file* berupa *plain text*) yang diupload oleh *user*. Dokumen yang di-*upload* otomatis akan tersimpan dalam *database* sistem.

Gambar 2. Skema aliran data pada sistem

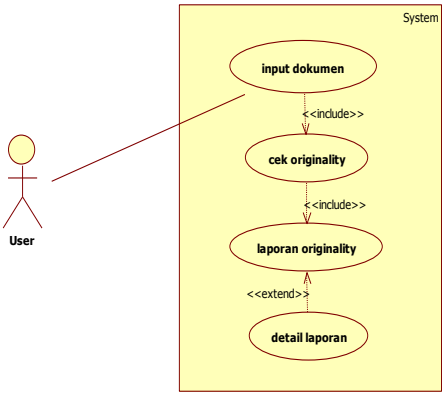
Selanjutnya dokumen akan melewati tahap *preprocessing* menggunakan *text mining*. Tahap ini terdiri dari *tokenizing* (memecah kalimat menjadi potongan kata sekaligus mengubah isi dokumen menjadi huruf kecil/*lowercase*), *filtering* (membuang *stopword*/ kata yang tidak deskriptif), dan *stemming* (mengembalikan setiap kata ke bentuk dasarnya).

Setelah itu, dokumen hasil *preprocess* akan melewati tahap *processing* menggunakan algoritma Rabin Karp. Tahap ini adalah tahap pencocokan dokumen. Tahap ini terdiri dari *parsing K-gram* (memecah *string* ke dalam potongan *substring* sebanyak *k*), *hashing* (mengubah *substring k* ke dalam nilai *hash*), dan *string matching* (pencocokan *string* hasil *hashing*).

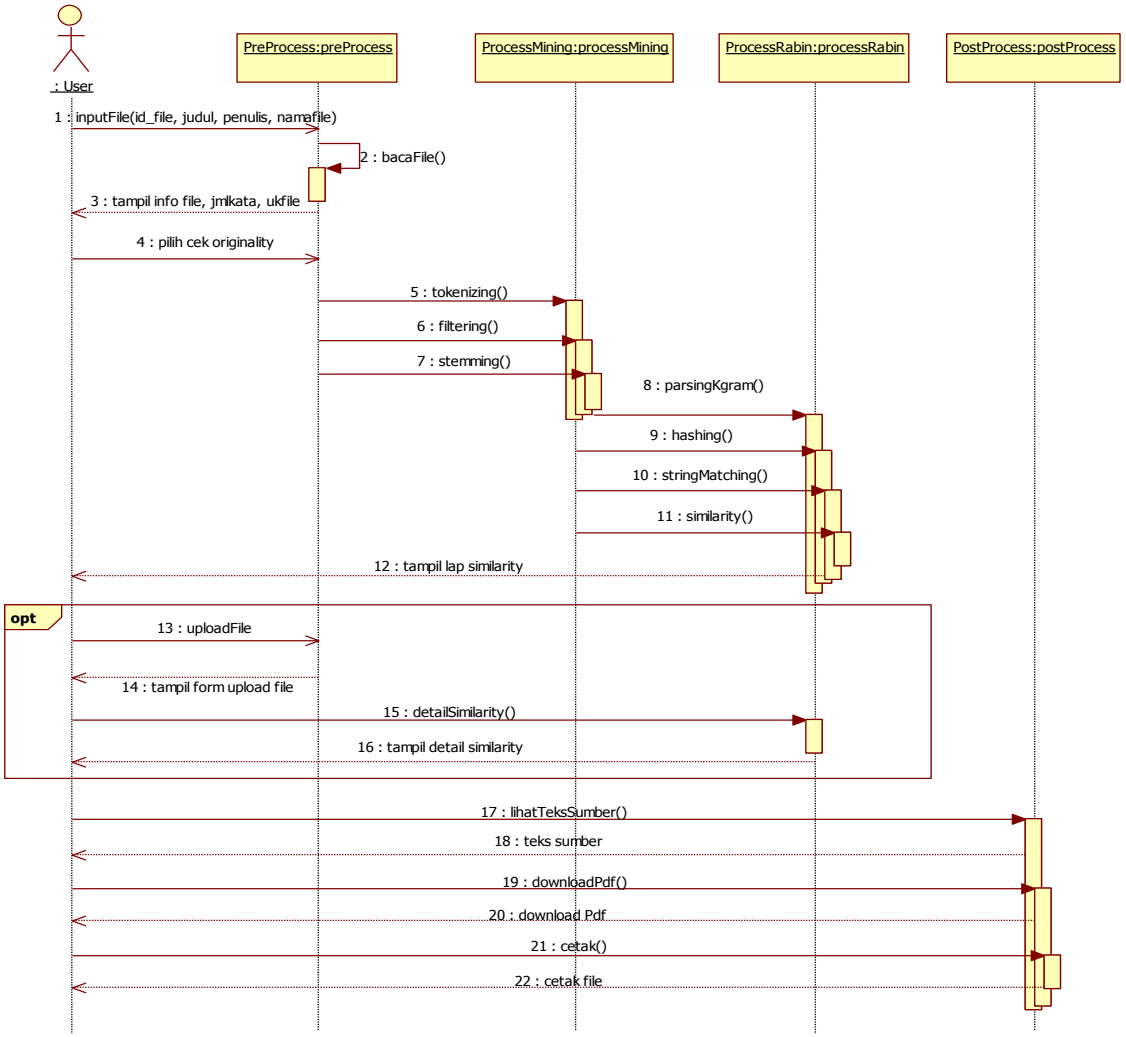
Hasil pencocokan *string* kemudian dihitung nilai *similarity* nya menggunakan *Dice's Similarity Coefficient*, yang kemudian nilai tersebut diubah ke dalam bentuk prosentase *similarity*.



Gambar 3. Arsitektur sistem



Gambar 4. *Use case diagram*



Gambar 5. *Sequence diagram*

III. HASIL PENELITIAN

A. *Hasil Uji Berdasarkan Banyaknya Konten File*

TABEL I berikut adalah informasi *file* yang akan digunakan untuk pengujian pada TABEL II dan TABEL III. *File* uji terdiri dari empat buah *file* yang masing-masing telah dilakukan penghapusan sebanyak 15%.

TABEL I. INFORMASI *FILE* UJI

No	Id	Nama	Kata	Ukuran (byte)
1	152	11_File_uji_14_kata.txt	14	95
2	153	12_File_uji_28_kata.txt	28	194
3	154	13_File_uji_42_kata.txt	42	316
4	155	14_File_uji_56_kata.txt	58	431

TABEL II. HASIL UJI BERDASARKAN BANYAKNYA KONTEN *FILE*

No Uji	File 1	File 2	Similarity (%)	Waktu (s)
147	11_File_uji_14_kata.txt	10_File_uji_14_kata.txt	100	0.213548
148	12_File_uji_28_kata.txt	10_File_uji_14_kata.txt	69	-0.139719
149	13_File_uji_42_kata.txt	11_File_uji_14_kata.txt	50	0.387906
150	14_File_uji_56_kata.txt	12_File_uji_14_kata.txt	44	0.335834

Dari TABEL II dapat diketahui bahwa semakin banyak konten *file*, maka semakin lama waktu proses.

B. Hasil Uji Tanpa Menggunakan Stemming

TABEL III. HASIL UJI TANPA MENGUNAKAN *STEMMING*

No Uji	File 1	File 2	Similarity (%)	Waktu (s)
151	11_File_uji_14_kata.txt	10_File_uji_14_kata.txt	100	0.038528
152	12_File_uji_28_kata.txt	10_File_uji_14_kata.txt	67	0.057912
153	13_File_uji_42_kata.txt	11_File_uji_14_kata.txt	46	0.061899
154	14_File_uji_56_kata.txt	12_File_uji_14_kata.txt	39	0.084752

Dari TABEL III dapat diketahui bahwa tanpa menggunakan stemming akan mempercepat waktu proses, tetapi akurasi yang *similarity*-nya rendah.

C. Hasil Uji Modulo pada Algoritma Rabin Karp

File yang digunakan untuk pengujian *modulo* dan *k-gram* adalah *file* 1 dan 2 pada TABEL I.

TABEL IV. HASIL UJI MODULO

No Uji	Kgram	Modulo	Similarity (%)	Waktu (s)
159	1	13	523.944	0.871916
161	1	23	523.944	0.153122
163	1	43	523.944	0.175531
164	1	71	523.944	0.209635
165	1	101	523.944	0.093315
166	1	151	523.944	0.151603
167	1	173	523.944	0.125665
168	1	251	523.944	0.177291

Berdasarkan TABEL IV dapat disimpulkan bahwa modulo tidak berpengaruh pada prosentase *similarity*, tapi berpengaruh pada waktu proses.

D. Hasil Uji k-gram pada Algoritma Rabin Karp

TABEL V. HASIL UJI K-GRAM

No Uji	Kgram	Modulo	Similarity (%)	Waktu (s)
172	1	101	523.944	0.045441
173	2	101	128.571	0.105267
174	3	101	71.0145	0.101174
175	4	101	69.1176	0.020179
176	5	101	68.6567	0.150793
177	6	101	68.1818	0.183669
178	7	101	67.6923	0.235006

Berdasarkan TABEL V dapat diketahui bahwa semakin kecil *k-gram*, akurasi *similarity*-nya semakin tinggi.

IV. KESIMPULAN

Berdasarkan percobaan-percobaan yang telah dilakukan, dapat ditarik kesimpulan bahwa sistem dalam membandingkan *file* memberikan hasil berupa prosentase *similarity*. Selain itu terdapat beberapa faktor yang mempengaruhi performa algoritma Rabin Karp, diantaranya banyaknya konten sebuah file akan memperpanjang waktu prosesnya (*running time*), *stemming* dan *preprocessing* membuat waktu proses cenderung lebih lama tetapi tingkat akurasi *similarity*nya lebih tinggi, nilai *modulo* berpengaruh pada waktu proses, dan semakin kecil *k-gram* akan menghasilkan akurasi nilai *similarity* yang lebih baik.

Penelitian lebih lanjut, diharapkan sistem dapat membandingkan *file* uji dengan semua *file* sumber pada *database* sistem. Jika masih menggunakan algoritma Rabin Karp, lebih dibenahi pada tahap *stemming* agar hasil pencocokannya lebih akurat. Diharapkan sistem dapat mendeteksi sinonim dan berbagai bentuk plagiat, serta dapat mengkonversi dokumen teks dengan ekstensi lain seperti .doc dan .pdf untuk kemudahan pengguna.

DAFTAR PUSTAKA

[1] R. G. V. Lukashenko and J. Grundspenkis, "Computer-based plagiarism detection methods and tools: an overview," in *ACM: CompSysTech: Computer Systems and Technologies*, New York, 2007.

[2] R. S. Chillar and B. Kochar, "RB-Matcher: String Matching Technique," *Rem (Text),* vol. 234567, no. 11, p. 3, 2008.

[3] D. Sugono and dkk, Kamus Besar Bahasa Indonesi (KBBI), http://pusatbahasa.kemdiknas.go.id/kbbi, 1997.

[4] B. Gipp and N. Meuschke, "Citation pattern matching algorithms for citation-based plagiarism detection: greedy citation tiling, citation chunking and longest common citation sequence," *Proceedings of the 11th ACM Symposium on Document Engineering (DocEng2011),* pp. 249--258, 2011.

[5] B. Zaka, "Theory and Applications of Similarity Detection Techniques," *Graz University of Technology,* 2009.

[6] S. Kosinov, "Evaluation of n-grams conflation approach in text-based information retrieval," in *8th String Processing and Information Retrieval Symposium (SPIRE 2001)*, Canada, Computing Science Department, 2001, pp. 136--142.

[7] R. Feldman and J. Sanger, The text mining handbook: advanced approaches in analyzing unstructured data, Cambridge: Cambridge University Press, 2007.

[8] C. Triawati, "Metode Pembobotan Statistical Concept Based untuk Klastering dan Kategorisasi Dokumen Berbahasa Indonesia," Institut Teknologi Telkom, Bandung, 2009.

[9] R. J. Mooney, "CS 391L: Machine Learning Text Categorization," University of Texas, Austin, 2006.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to algorithms, USA: MIT Press, 2001.

[11] A. Atmopawiro, Pengkajian dan Analisis Tiga Algoritma Efisien Rabin-Karp, Knuth-Morris-Pratt, dan Boyer-Moore dalam Pencarian Pola dalam Suatu Teks, Bandung: STEI ITB, 2006.

[12] J. M. Elchison, "Cedarville University - CS3410-Algorithm," 19 November 2004. [Online]. Available: http://www.cedarville.edu/personal/personalpages/shomper/cs3410_web/resources/rabin_karp_matching.ppt. [Accessed 5 November 2012].